Finite Volume Droplet Trajectories for Icing Simulation

T.C.S. Rendall, C.B. Allen

Department of Aerospace Engineering, University of Bristol, Bristol, BS8 1TR, U.K.

Abstract

During a Lagrangian icing simulation a large number of droplet trajectories are calculated to determine the water catch, and as a result it is important that this procedure is as rapid as possible. In order to arrive at a method with minimum complexity, a finite volume representation is developed for streamlines and extended to incorporate the equations of motion for a droplet, with all cells being crossed in a single timestep. However, since cells vary greatly in size, the method must be implicit to avoid an awkward stability restriction that would otherwise degrade performance. An implicit method is therefore implemented by carrying out iterations to solve for the crossing of each CFD cell, so that the droplet motion is tightly coupled to the underlying flow and mesh. By crossing every cell in a single step, and by using the mesh connectivity to track the droplet motion between cells, any need for costly searches or containment checks is eliminated and the resulting method is efficient. The implicit system is solved using functional iteration, which is feasible for the droplet system (which can be stiff) by using a particular factorisation. Stability of this iteration is explored and seen to depend primarily on the maximum power used in the empirical relationship for droplet drag coefficient $C_D = C_D(R_e)$, while numerical tests confirm the theoretical orders of accuracy for the different discretisations. Final results are validated against experimental and alternative numerical water catch data for a NACA 23012 aerofoil.

Email address: thomas.rendall@bristol.ac.uk (T.C.S. Rendall)

1. Introduction

Calculation of droplet or solid particle trajectories through a CFD mesh is a problem that arises across a wide range of disciplines, ranging from icing simulation, to combustion modelling, chemical engineering, fire sprinkler systems and haemodynamics. Modern techniques apply CFD to calculate the flow field surrounding droplets, and then integrate Newton's second law to determine the trajectory. However, this means using the flow field information (which usually consists of just the velocity, but it might also include temperature or other flow parameters), which is Eulerian in nature, within a Lagrangian calculation, and hence requires a method of tracking.

Water catch calculations used in icing analysis may be conducted in either a Lagrangian (Ruff and Berkowitz, 1990) or Eulerian (Beaugendre et al., 2003; Bourgault et al., 1997) frame of reference. The Lagrangian approach is more common, and requires droplet trajectories to be calculated, whereas the Eulerian approach solves transport equations to determine the water catch on the surface of the object directly. For small droplets an Eulerian frame of reference is likely to be superior, but for larger droplets the assumption of a volume weighted droplet quantity breaks down. Larger droplets are also likely to exhibit behaviour that is more easily expressed in a Lagrangian frame, such as splashing, and regulatory bodies will soon require certification under these conditions. This work describes methods to perform trajectory calculations for the Lagrangian approach.

For these computations the form of the underlying flow solution is very important. If it is a potential flow solution (Ruff and Berkowitz, 1990), which is often used for a rapid analysis, then it is straightforward to evaluate the flow velocity at any arbitrary position from the velocity potential. In comparison, a CFD volume mesh (Potapczuk, 1992) requires the droplet to be tracked as it passes through the mesh from cell to cell. An interpolation within each cell for the flow velocity may be used for increased accuracy. It is expected that as the development and usage of CFD progresses, together with a need to anaylse more elaborate shapes, tracking droplets through a volume mesh will become more common in icing calculations.

Tracking a massless particle (to give a streamline) through a finite-volume mesh is

in principle straightforward, but requires an efficient method for finding which CFD cell in a mesh contains the particle, as many millions of droplets may ultimately need to be tracked. The level of complexity is increased by introducing the droplet equations of motion, as these can come with timestep stability restrictions, and for efficiency it is necessary to resolve these numerical limitations with the method of tracking. It is this combination of tracking and numerical integration that is of principal interest here; for the tracking method to work, every cell must be crossed in an integer number of timesteps (here, always one), which means the integrator must be able to handle any timestep, even if the system is stiff.

This work starts by describing the simplest possible way to represent streamlines on a finite-volume mesh, before incorporating the equations of motion to give a trajectory. The final result is a method that is fast, adaptive to the volume mesh resolution and without timestep restrictions. The basic methods are first and second order accurate in space and time, but these orders may be increased at the expense of more restrictive stability bounds if needed, and alternative integrators can also be fitted in the face intersecting framework.

2. Implicit Mesh Dependent Streamlines and Droplet Trajectories

The nonlinear ordinary differential equations that describe the trajectory of a water droplet are usually solved by a numerical integration technique. Many methods use an explicit single or multi-stage approach, with Runge-Kutta and predictor/corrector techniques being common, although for the stiff equations that result from low mass particles integrators designed for stiff systems are preferred. Nonlinearity arises both from the dependence of the droplet drag coefficient on the Reynolds number, and the link to the dynamic pressure of the relative flow speed between the droplet and the air.

The choice between an explicit and an implicit integration scheme is fundamental (Kim and Elangovan, 1986). An explicit scheme imposes a stability restriction on the timestep, while an implicit scheme is left largely unfettered in this regard and the timestep only needs to be guided by accuracy considerations. An implicit droplet scheme is comparatively much simpler than an implicit CFD scheme, and requires in the simplest case only the solution of a single nonlinear equation for each timestep.

The cost and difficulty of simulating large numbers of trajectories of small droplets has been noted previously, both by Kim and Elangovan (Kim and Elangovan, 1986) who suggested a switch towards semi-implicit schemes, and Caruso (Caruso, 1993) who advocated parallel processing of trajectories.

2.1. Streamlines

The operations required to calculate a droplet trajectory are closely related to those used to find a streamline, so for simplicity this case shall be considered first. Many streamline methods involve using cell neighbours to compute velocity interpolations, and then integrate the velocity vector to find the streamline, usually taking multiple integration steps within a cell. Although the apparent accuracy of this approach may appear good (with a large number of steps in a cell giving a smoothly curving streamline) the actual accuracy still depends on the underlying flow velocity. The flow equations are typically solved using a finite volume discretisation, and this uses a constant velocity within a cell (often with a gradient based face reconstruction procedure), so although an attempt to increase the accuracy of the streamline beyond that of the solver may be visually pleasing it is not always based on the underlying discretisation. Further, it is clear that there exists a consistent 'finite-volume' (FV) streamline that uses the cell velocities and no other information. It is this fundamental FV streamline description that presents an efficient and accurate way of tracing streamlines and trajectories.

A FV streamline consists only of straight line segments pointing along the direction of the flow velocity in each cell (note that if the containment cell velocity is taken, the method is first order), while points along the streamline are defined by the intersections of these segments with cell faces, as illustrated in figure 2. Importantly FV streamlines do not require integration of any differential equation to be traced out. They are defined only by the discrete flow solution and the mesh, i.e. they are geometrical in nature rather than mechanical. Streamlines may also be considered as the trajectories of massless particles; since these particles have no mass, they accelerate instantaneously to match the flow velocity in any cell of the mesh. This instantaneous acceleration leads to the discontinuities in velocity between cells.

The process of finding a streamline in this manner is simple and uses only the

connectivity information already required by the flow solver. Two kinds of connectivity are required: (1) edge-cell indexing and (2) cell-edge indexing. Edge-cell indexing allows the neighbouring cells (or boundary conditions) of a particular edge to be found, while cell-edge indexing allows the neighbouring edges of a particular cell to be found. The data required for edge-cell connections are already used by the flow solver, so no work is required here, while cell-edge connectivity may be determined exactly from the edge-cell data in a simple preprocessing step.

With this connectivity in hand, the streamline is found by first placing a seed point within a known cell. Intersection points of the velocity ray from this seed point to the infinite planes of all the cell faces are then found, with the nearest one that lies in the positive direction along the velocity vector finally being chosen as the correct point. Once the exit edge has been found, the next seed point is this intersection point, and the next velocity ray is determined by the velocity in the cell on the other side of the edge. Alternatively, the other side of the edge may be a boundary, in which case the intersection point is recorded and the process stops. Exactly one step is therefore required per cell, and the cost of finding a streamline is proportional to the mesh size; the only searching that is needed takes place over the boundary edges of a cell and since this is usually a small number (3-6) that is roughly constant within the mesh this does not add a significant searching expense. Visually it can be seen that a number of postprocessing tools, commercial and in-house, use this technique.

The complete process for a streamline is:

1. Pick a starting point in a known cell
2. Using the cell-edge connectivity calculate all intersections between the velocity vector for that cell with the faces of that cell
3. Select the nearest intersection point in the positive direction along the velocity vector as the next point for the streamline. Alternatively, select negative intersections for the upwind streamline
4. Use the edge-cell connectivity to locate the cell on the other side of this edge, or terminate if this is a boundary edge, and then move into this cell

In a small number of cases the intersection between the velocity vector and a face

may be extremely flat. In this case, roundoff errors can cause intersections in the positive direction to be mistakenly found as negative. To escape this error the method of (Haselbacher et al., 2007) is applied; the actual point of intersection is left unchanged and the streamline moved into the cell on the other side of the edge in question.

Overall this method achieves first order accuracy if the cell flow velocity is used to intersect the faces. This can be improved to second order if a gradient extrapolation is used to the face, so that

$$\mathbf{v}_f = \mathbf{v}_c + \nabla \mathbf{v} \cdot \mathbf{d} \tag{1}$$

where $\mathbf{d}$ is the vector from the cell centre to the face intersection point. The consistent velocity to use to intersect the faces of the cell is then the average of this extrapolated value and the value at the previous face intersection, rather than the cell velocity alone. To verify that this technique does improve spatial accuracy, a potential flow vortex was placed in a sequence of refined triangular meshes (as later described in section 5.1), and a semi-circular streamline calculated. Knowing the start location the error at the end point could be found, and the accuracy inferred using a regression line. The constant velocity approach reached an order of accuracy of 0.97 while applying a gradient extrapolation improved this to 2.4 (a result in excess of 2 is fortuitous and likely to result from coincidental cancellation errors).

2.2. Droplet Trajectories

A range of different methods have been suggested both for locating a particle, and also for tracking its path through the mesh (Zhou and Leschziner, 1999; Chorda et al., 2002; Haegland et al., 2002; Schafer and Breuer, 2002; Sadarjoen et al., 1994). The method of interest here follows from work by Haselbacher et al. (2007), Kuang et al. (2008) and Macpherson et al. (2009) and builds on the face-intersecting approach taken for finding streamlines. This method has been presented subsequently with modifications aimed at addressing inconsistencies for nearly degenerate intersections, and further instances in the literature may exist. Haselbacher et al. (2007) and Kuang et al. (2008) considered particle trajectories, but not how to use an integration technique in conjunction with a process for face intersection. Indeed, it would appear that the possibility of dealing with the tracking process and integration of the equation of motion

simultaneously in a single step has escaped attention, with methods tacitly assuming this would not be worthwhile. Different work by Nielson and Jung (1999), and later Kipfer et al. (2003) explored using local analytical integration, which was then combined with face intersection to give points along the trajectory. However, whilst feasible for streamlines and certain droplet models, this is not sufficiently general as the equations of motion are not always integrable.

The work presented here extends the face intersection method by building an iterative implicit integration routine into the intersection process, allowing any cell to be traversed in a single step with the equations of motion being solved simultaneously. In contrast to streamlines, trajectories are mechanical objects that require integration of Newton's second law

$$\frac{d\mathbf{u}}{dt} = \frac{\mathbf{f}}{m} + \mathbf{g} = F(\mathbf{v} - \mathbf{u}) + \mathbf{g} \tag{2}$$

for their calculation, where $\mathbf{v}$ is the flow velocity in the cell, $\mathbf{u} = \mathbf{u}(t)$ is the droplet velocity through the cell and $\mathbf{f}(t)$ is the force. The mass m will from here on be assumed constant, though a variable mass can be accommodated providing there is an equation to determine it, and could also be included in the implicit framework given later.

Droplet parameters are

$$A = \pi r^2; \qquad m = \rho_w \frac{4}{3}\pi r^3; \qquad C_D = C_D(Re); \qquad Re = \frac{\rho\|\mathbf{v} - \mathbf{u}\|2r}{\mu} \tag{3}$$

where ρ is the density of air, ρ_w the density of water, r the droplet radius, A the reference area, m the mass, μ the viscosity of air and Re the Reynolds number based on the diameter. The scalar function F is given by

$$F = \frac{C_D \frac{1}{2}\rho\|\mathbf{v} - \mathbf{u}\|^2 A}{m\|\mathbf{v} - \mathbf{u}\|} = \frac{3C_D\rho\|\mathbf{v} - \mathbf{u}\|}{8\rho_w r} \tag{4}$$

In this work the drag coefficient for $Re < 3500$ is found from (Gent et al., 2003)

$$\frac{C_D Re}{24} = 1 + 0.197Re^{0.63} + 2.6Re^{1.38} \tag{5}$$

or for $Re > 3500$ from

$$\frac{C_D Re}{24} = 1.0 + 0.197Re^{0.63} + 2.6 \times 10^{-4} Re^{1.38} \tag{6}$$

Putnam's relation (Putnam, 1961)

$$\frac{C_D Re}{24} = 1 + \frac{Re^{\frac{2}{3}}}{6} \tag{7}$$

may also be used. The advatage of equation (7) it that it permits analytical integration, making it invaluable for comparison purposes, but in general more accurate or case-specific non-integrable results such as equation (5) (or others) are used.

The velocity of the flow $\mathbf{v}$ and its gradient are known from the CFD cells of the solution, while $\mathbf{u}$ is unknown. Importantly the time spent within the CFD cell, Δt, is also unknown. This gives four unknowns (three components of $\mathbf{u}$ plus Δt) but only three equations (the equations of motion in x,y and z). The additional equation is

$$\Delta t = g(\mathbf{u}, \mathbf{p}) \tag{8}$$

This expresses the time of flight across a particular CFD cell as a function of the particle velocity in the cell and the point $\mathbf{p}$ where the particle enters the cell. The function g is the geometric tool that calculates where the particle exits the cell, given its velocity and entry location. This means a scheme is required both to advance the droplet trajectory across a single cell and calculate the timestep, as this cannot be specified in advance. Developing a face intersecting scheme has two big mutually advantageous properties in this regard:

- It adheres to the philosophy that an efficient representation of a trajectory should consist of the points along the trajectory where it intersects cell faces. This keeps the number of points along a trajectory proportional to the mesh size
- The size of the timestep automatically tracks the size of the CFD cells, so timestep adaptation is implicit via the CFD mesh

It is important the scheme is unconditionally stable, as the timestep will vary greatly and cannot be known in advance. It is also preferable that the scheme shows L-stability (Kim and Elangovan, 1986) (unconditional stability plus correct stiff decay behaviour) in order to avoid trajectory oscillations for low mass droplets. Stiff decay can be understood by considering the trial equation $\dot{y} = \lambda y$. Discretising this with the backward Euler method $y^{n+1} = y^n + \lambda \Delta t y^{n+1}$ leads to $\frac{y^{n+1}}{y^n} = \frac{1}{1-\lambda\Delta t}$, while using a mid-point

gives $y^{n+1} = y^n + \frac{\lambda\Delta t}{2}(y^{n+1} + y^n)$ and therefore $\frac{y^{n+1}}{y^n} = \frac{2+\lambda\Delta t}{2-\lambda\Delta t}$. When λ is large and negative the limit amplification factor is different; for the backward Euler scheme zero is obtained, while the mid-point method leads to -1. The correct behaviour of the differential equation is rapid exponential decay, so a limit amplification factor of anything other than zero is incorrect. Schemes not satisfying this condition can produce physically incorrect oscillations for low mass droplets despite being stable.

3. Numerical Schemes

The simplest method of solving this system numerically is a first order explicit (forward Euler) method. This may be constructed by assuming that within the cell $\mathbf{f}$ is constant with $\mathbf{u} = \mathbf{u}^{n-1}$. For this approach, the nearest intersection is required between the droplet velocity vector and all cell faces. This reveals an obstacle in that the velocity vector depends on the integration of the acceleration (which depends in turn on the velocity) and on the timestep (which depends on the velocity). Neither of these quantities are known in advance for the cell. Hence, it is clear that if face intersections are to be used the forward Euler scheme encounters difficulties.

A solution would be to lag the droplet velocity behind the flow velocity by one face intersection, but this would introduce an extremely undesirable phase lag between the droplet velocity and force, and this would be particularly poor for the larger timesteps expected here. A constant, much smaller, timestep could be used to alleviate this, and the restriction to require a trajectory to consist of face intersection points dropped. However, this gives a result that may require an inefficiently small timestep and a searching operation at the end of every timestep to check for face crossings. It has been observed (Kim and Elangovan, 1986) that for low droplet masses the stability restriction on the timestep for such schemes can pose a serious efficiency problem.

Explicit methods are a sensible choice providing the timestep size is efficient and any phase lag is not significant; when this is not the case a switch to an implicit scheme needs to be considered. From the range of stiff integrators available backward difference formulae (BDF) are some of the simplest and most effective implicit methods. The 1^{st} and 2^{nd} order schemes (BDF1 and BDF2) exhibit L-stability, which is very helpful for the stiff system seen for low mass droplets. As shown by Darmofal and Haimes

(1996) (although in the context of unsteady streamlines), even for pure streamline calculations backward differencing is competitive compared to alternative integrators.

3.1. First Order Scheme

Discretising Newton's second law for the droplet trajectory, and adopting the nomenclature indicated in figure 1, using BDF1 (backward Euler) gives

$$\alpha m\mathbf{u}^n + \beta m\mathbf{u}^{n-1} = \mathbf{f}\left(\mathbf{v},\mathbf{u}\right) + m\mathbf{g} = mF(\mathbf{v}-\mathbf{u}) + m\mathbf{g} \tag{9}$$

with $\alpha = \frac{1}{\Delta t}$ and $\beta = \frac{-1}{\Delta t}$.

An implicit expression can be constructed at n to give the velocity of the droplet exiting the cell, $\mathbf{u}^n$, as

$$\mathbf{u}^n = \frac{F\mathbf{v}^n - \beta\mathbf{u}^{n-1} + \mathbf{g}}{\alpha + F} \tag{10}$$

where $\mathbf{g}$ is the acceleration due to gravity and F is a function that relates the relative velocity between the droplet and the flow $\mathbf{v}^n - \mathbf{u}^n$ to the force acting on the droplet. This BDF1 method is unconditionally stable.

The iteration used to solve this is

$$\mathbf{u}^{n,i+1} = \frac{F^{n,i}\mathbf{v}^n - \beta\mathbf{u}^{n-1} + \mathbf{g}}{\alpha + F^{n,i}} \tag{11}$$

which is continued until consecutive iterations in i product a negligible change. Stability of this functional iteration can be linked to $C_D = C_D(R_e)$ and will be considered in section 4.

The airflow velocity at the interface between cells c and $c+1$ can use a central difference (consistent with many flow solvers)

$$\mathbf{v}^n = \frac{1}{2}\left(\mathbf{v}_c + \mathbf{v}_{c+1}\right) \tag{12}$$

This expression requires the current cell c and the 'next' cell $c+1$ to be known. $c+1$ is only known once the implicit iterations have converged, so it can change its value between consecutive iterations. Before the iterations begin (and while $c+1$ is therefore undefined) $\mathbf{v}^n = \mathbf{v}_c$ is used as a starting guess. However, with strong variations in timestep a central scheme of this type may reduce to a first order scheme. In order to alleviate this a gradient extrapolation to the face identical to that used for finding

streamlines may be used, as given in equation (1). This results in a reliably second order method even if the timesteps vary between integration steps.

Equation 11 represents fixed point iteration on $\mathbf{u}^n$. An initial guess for $\mathbf{u}^n$ is that it is equal to the flow velocity in the cell (the added inertia means that for larger mass droplets it may be faster to guess $\mathbf{u}^n = \mathbf{u}^{n-1}$, although the benefit is probably slight), and after the nearest face intersection is found this gives Δt. The nearest face intersection is found using a mid-cell velocity $\mathbf{u}^{n-\frac{1}{2}} = \frac{1}{2}\left(\mathbf{u}^n + \mathbf{u}^{n-1}\right)$. A new $\mathbf{u}^n$ is then found using the updated Δt and the process iterated to convergence, typically requiring 2-5 cycles for completion to a good accuracy. The iteration steps between integrating the equations of motion and finding the nearest face intersection of the mid-cell velocity ray (which directly implies Δt).

When the mass of the droplets is very low, $F \to \infty$, therefore $\mathbf{u}^n \approx \mathbf{v}^n$ and the droplet velocity is nearly equal to the flow velocity. When the mass is large $F \approx 0$ and so $\mathbf{u}^n \approx \mathbf{u}^{n-1} + \Delta t \mathbf{g}$ and the droplet velocity is changed only by gravity over the timestep; heavy droplets are unaffected by aerodynamic forces.

3.2. Second Order Scheme

A method second order accurate in velocity may also be built. Again using a backward difference, an implicit expression at time level n is

$$\alpha m\mathbf{u}^n + \beta m\mathbf{u}^{n-1} + \gamma m\mathbf{u}^{n-2} = \mathbf{f}\left(\mathbf{v}, \mathbf{u}\right) + m\mathbf{g} = mF(\mathbf{v} - \mathbf{u}) + m\mathbf{g} \tag{13}$$

where α, β, γ are the differencing coefficients

$$\alpha = \frac{2\Delta t_c + \Delta t_{c-1}}{(\Delta t_c + \Delta t_{c-1})\Delta t_c} \tag{14}$$

$$\beta = \frac{-(\Delta t_c + \Delta t_{c-1})}{\Delta t_c \Delta t_{c-1}} \tag{15}$$

$$\gamma = \frac{\Delta t_c}{(\Delta t_c + \Delta t_{c-1})\Delta t_{c-1}} \tag{16}$$

These allow for a variable timestep, as this is expected to vary significantly between steps. Re-arrangement gives

$$\alpha\mathbf{u}^n + \beta\mathbf{u}^{n-1} + \gamma\mathbf{u}^{n-2} = F\left(\mathbf{v}^n - \mathbf{u}^n\right) + \mathbf{g} \tag{17}$$

$$\mathbf{u}^n = \frac{F\mathbf{v}^n - \beta\mathbf{u}^{n-1} - \gamma\mathbf{u}^{n-2} + \mathbf{g}}{\alpha + F} \tag{18}$$

The implicit iteration is then

$$\mathbf{u}^{n,i+1} = \frac{F^{n,i}\mathbf{v}^n - \beta\mathbf{u}^{n-1} - \gamma\mathbf{u}^{n-2} + \mathbf{g}}{\alpha + F^{n,i}} \tag{19}$$

3.3. Third Order Scheme

A third order scheme (BDF3) may be constructed in a similar manner. To do this, the difference coefficients are extended to third order and a quadratic is fitted through the droplet velocities at n, $n-1$ and $n-2$ (n is the time level under iteration). This quadratic may then be used to evaluate the velocity at the centre of the current volume, rather than using the simpler mid-cell average velocity applied for BDF1 and BDF2. This velocity ray is intersected with the faces of the cell in the same manner as for BDF1/BDF2 to give the timestep, but then the final position is calculated using integration of the quadratic velocity interpolation, rather than being taken as the face intersection of the ray.

It is not necessary to solve for the intersection of the cubic trajectory with the cell faces exactly; it is only necessary to calculate Δt to the same accuracy as the discretisation of the equation of motion, which the quadratic interpolation provides. This means higher-order schemes do not need to solve for higher-order polynomial-face intersections, which would be error-prone. The approach of Nielson and Jung (1999) and Kipfer et al. (2003) would be similar to using polynomial-face intersections, but with the actual analytical solution.

Figure 3 shows the regions of stability of the three schemes when applied to the test equation $\dot{y} = \lambda y$, where the complex variable is $\lambda\Delta t$ and Δt is constant. These curves were generated by finding the root of the characteristic polynomial having the greatest magnitude in the complex plane and plotting the contour where this crossed unity (Ascher and Petzhold (1998)). This shows that in practice BDF3 is not suitable here as small regions of instability exist to the left of the imaginary axis (see figure 3), and these will normally catch at least some of the trajectories, leading to failure. This makes BDF3 less useful than BDF1/2 here. However, the ability to use face intersections is not intrinsic to the BDF methods used here or their order of accuracy,

and any other unconditionally stable integration could be used. The concept of using a polynomial interpolation followed by the same face intersection procedure would apply to any alternative integrator, and as shown in section 5.2, will preserve the order providing the interpolated velocity is used in the face intersection process.

4. Convergence of the Iteration

Consider the model equation

$$\dot{u} = K(v-u)f(v-u) \tag{20}$$

where $f = |v-u|^{\alpha_p}$ is a function that mimics the effect of the Reynolds number on the drag coefficient, as well as part of the dynamic pressure. Writing a functional iteration for BDF1 or BDF2 to find u^n gives

$$\alpha u^{n,i+1} = -\beta u^{n-1} - \gamma u^{n-2} + K(v^n - u^{n,i})f(v^n - u^{n,i}) = \alpha G_1(u^{n,i}) \tag{21}$$

Equation (21) shows if $K \to \infty$ then $u^{n,i+1}$ does not approach v^n. When K is large (system is stiff) this type of iteration is well known to fail (Shampine and Gear, 1979). However, if part of the dynamic pressure is handled implicitly, so that

$$\alpha u^{n,i+1} = -\beta u^{n-1} - \gamma u^{n-2} + K(v^n - u^{n,i+1})f(v^n - u^{n,i}) \tag{22}$$

Then

$$u^{n,i+1} = \frac{-\beta u^{n-1} - \gamma u^{n-2} + K v^n f(v^n - u^{n,i})}{\alpha + K f(v^n - u^{n,i})} = G_2(u^{n,i}) \tag{23}$$

This shows that if $K \to \infty$ then $u^{n,i+1} \to v^n$, which is the correct behaviour in the stiff case where the droplet velocity must approach the flow velocity. This is a useful remedial adjustment to the iteration, which would otherwise need to be swapped for a Newton scheme (see for example Kipfer et al. (2003) for a Newton method applied to streamlines). The inclusion of the extra time levels from BDF2 does not alter the stiff decay towards v^n.

Differentiating

$$\frac{dG_2}{du^{n,i}} = G_2' = \frac{(-\beta u^{n-1} - \gamma u^{n-2} - \alpha v^n)K f'}{(\alpha + Kf)^2} \tag{24}$$

Considering the requirement for convergence of functional iteration that $|G_2'| < 1$, this is satisfied for both $K = 0$ and $K \to \infty$. Between these bounds

$$G_2' = \frac{(-\beta u^{n-1} - \gamma u^{n-2} - \alpha v^n) K \alpha_p \left|v^n - u^n\right|^{\alpha_p - 1}}{(\alpha + K \left|v^n - u^n\right|^{\alpha_p})^2} \tag{25}$$

It is now simplest to consider the case where $u^n \approx u^{n-1} \approx u^{n-2}$ with a single value of the timestep. This means that for BDF1

$$G_2' = \frac{-(v^n - u^n) k \alpha_p \left|v^n - u^n\right|^{\alpha_p - 1}}{(1 + k \left|v^n - u^n\right|^{\alpha_p})^2} \tag{26}$$

with $k = \Delta t K$. For BDF2 the result is larger

$$G_2' = -\frac{3}{2} \frac{(v^n - u^n) k \alpha_p \left|v^n - u^n\right|^{\alpha_p - 1}}{(1 + k \left|v^n - u^n\right|^{\alpha_p})^2} \tag{27}$$

For the model of C_D used $\alpha_p = 1.38$ or $\alpha_p = 0.66$ (for Putnam's C_D), so the largest power of velocity difference in the numerator is 1.38 whilst that in the denominator is 2×1.38, and convergence is ensured for large velocity differences. If the velocity difference is zero, then f' is undefined but clearly $G_2' = 0$ still.

If G_2' for BDF1 is plotted as a function of $\left|v^n - u^n\right|^{\alpha_p}$ at constant k and α_p, or as a function of k at constant $\left|v^n - u^n\right|^{\alpha_p}$ and α_p, in both cases a maximum is reached with a value of $\frac{\alpha_p}{4}$ (see figure 4 for $\alpha_p = 1$), or $\frac{3}{8}$ for BDF2. This implies that BDF1 cannot succeed if C_D depends on the Reynolds number to a power greater than three (the difference of one is because in the above f contains a proportional term from the dynamic pressure, so even if C_D was constant α_p would still be one), while BDF2 cannot succeed if this power exceeds $\frac{8}{3} - 1 = \frac{5}{3}$. In practice, empirical results rarely use powers above one (the range $-0.5 \to 0.5$ contains most models), so this restriction is unimportant. Experimentation for the example used in section 5.2 showed failure for a power of 3.8, which is likely to be higher than the expected value as no timestep happened to exactly hit the maximum of the curve in figure 4. On meshes with a wider range of timesteps and trajectories it would be expected for at least a few of the timesteps on some of the trajectories to fall close to the maximum, resulting in the observed limit being much closer to the theoretical one.

Hence, irrespective of stiffness, timestep and velocity deficit the iterations can be expected to converge for realistic C_D functions. In comparison, at a constant timestep

for BDF2 $G_1' = -k(f + v^n f' - u^n f')$ so $|G_1'|$ can only be less than unity if k is sufficiently small, which represents the well known timestep/stiffness limit.

5. Validation

5.1. Spatial Accuracy

The value of the flow velocity calculated at face intersection points controls the spatial accuracy of the method, as this is the link back to the underlying flow solution. Using a face average gives a method that is second order on a suitably smooth mesh; however, as the timesteps can vary significantly due to varying intersection paths, such a method can reduce back to a first order scheme. This motivates the use of equation (1).

Calculating $\nabla \mathbf{v}$ is the subject of much work in CFD. The most common approaches use either a Green-Gauss integration for each cell or a local least-squares fit using the cell and its neighbours. A Green-Gauss method often uses a node-centred mesh, as for simplicial meshes this allows the approach to recover the gradient of linear functions exactly, but cell-centred calculations are also common. Least-squares fitting always recovers a linear gradient exactly, but requires distance weighting for better conditioning and good accuracy near boundaries (Mavriplis (2003)). In this work, a Green-Gauss gradient has been used.

To verify the order of (1) for this application a potential flow vortex was placed in a square of side unity meshed using triangles, and a droplet trajectory was calculated emanating from a point near the core as shown in figure 5. The error between the calculated face velocity and the real face velocity found from potential flow was summed along all points to produce a RMS value, together with a RMS value for the timestep size, before linear regression was used to estimate the order of accuracy with grids using 3734, 8884, 14955, 29907 and 61147 triangles.

Plain averaging achieved first order accuracy (0.986), while the gradient based method achieved a range of results depending on the accuracy of the calculated gradient. With exact face values used in the Green-Gauss integral, second order was achieved (2.036), while use of linearly interpolated face values within the Green-Gauss

integral resulted in an accuracy of 1.149. This is consistent with the comments of Bonfiglioli and Leschziner (2001). It should be noted that many flow codes compute velocity gradients internally, and the trajectory routine could use these to interrogate the flow velocity to commensurate accuracy.

5.2. Time Accuracy

To confirm the temporal order of accuracy, tests have been performed to compare the numerical solution to an analytical solution. If Putnam's form (Kim and Elangovan, 1986) is chosen for $C_D = C_D(R_e)$ then the equations of motion are integrable, but still closely representative of a real calculation with an alternative C_D variation.

To make the comparison, a droplet was introduced into a uniform square grid of side unity. The airflow was set to 100ms^{-1} left to right with a droplet radius of 25μm and an initial droplet velocity of (-100,100)m/s, giving a tightly curved region followed by limiting behaviour towards the freestream, as shown by the example trajectory in figure 6(a). The velocity magnitude at the exit point was then computed and compared to the analytical solution to give an error ϵ. For BDF2/BDF3, the previous time levels (required to start the calculation) were found by evaluating the analytical velocity one or two timesteps back in time. If this is not done, asymptotic convergence on the finest grids is not seen owing to the errors introduced at initialisation.

Figure 6 shows the convergence on grids ranging from 80×80 to 2560×2560, with the 1^{st} order method showing an accuracy (measured with a least-squares line) of 0.999, the 2^{nd} order method an accuracy of 2.041 and the 3^{rd} order method an accuracy of 3.108, despite wide variations in timestep throughout the integrations. A repeat test was performed on a similar mesh but using a cosine rather than uniform grid spacing, with a maximum cell aspect ratio along the trajectory of 61. The orders of accuracy achieved for this case were 0.995, 2.038 and 3.071.

5.3. Water Catch Comparison

To validate against established water catch data, a NACA 23012 aerofoil at an incidence of 2.5^o was simulated using an inviscid finite volume CFD tool. The process began with iterations to detect the upper and lower impingement limits on the aerofoil, after which a rake of impinging droplets was initiated four chord lengths upstream,

as illustrated in figure 7. For each trajectory with a vertical start point y_0, a central finite difference was used to calculate the water catch. This required computing two additional trajectories starting from $y_0 + \delta$ and $y_0 - \delta$ to find the surface intersection locations, after which the water catch efficiency was found from $\beta = \frac{2\delta}{\Delta s}$, where Δs is the arclength separation of the two intersection points. This is the same process as described by Gent et al. (2003) and was selected here as it involves no intermediate smoothing procedure. In contrast, the method used in Lewice (Ruff and Berkowitz (1990)) uses a four-point least squares fit to $y_0 = f(s)$, which is then analytically differentiated to find β. Although results would be smoother, in this instance such a process might cloud comparisons.

The comparison in figure 9 with the experimental results of Papadakis et al. (2007) reveals that results follow the Lewice data reasonably closely. Differences between experiment and the Lewice predictions are attributable to splashing phenomena, which prevent all of the impinging water from being caught by the surface. The main difference between the results of Lewice and the current method (2^{nd} order temporal scheme with either a 1^{st} or 2^{nd} order velocity interpolation) is that the impingement limits are different, which may be attributed to the exclusion of any smoothing in post-processing and the use of a finite volume mesh; when intersecting straight line segments along the surface the exact tangency condition at the edges of the catch curve cannot be obtained. However, at these locations, the water catch is by definition low, and it is noticeable that the experimental impingement limits are more closely recovered by the current method, albeit in a truncated fashion. The catch calculation is as sensitive to the method used to find β as it is to the trajectory method itself. Figure 11 shows the influence of improving the accuracy of the underlying flow velocity value using equation (1). The differences are mainly towards the impingement limits, with little change in the maximum or its location.

6. Cost Comparisons

To enable a cost comparison to alternative methods two additional schemes were tested. The first (referred to as the 'optimal' method) was constructed by removing the tracking sections of the code completely; this is possible for the unit square test case

here as the flow is uniform, and the flow velocity can therefore be set without tracking droplets. A calculation of this type represents the cost of solving only the numerical problem, and no matter what approach is taken for finding the cell containment, the final performance cannot be better than achieved by this. In general this approach cannot be used, as arbitrary meshes and flow solutions demand a tracking method so that the flow velocity may be found from the containing cell.

A second method was also built which used an alternating digital tree (Bonet and Peraire, 1991) (termed the ADT method) to determine the cell containing the droplet at each timestep. This works by using the ADT to return a short list of cells in the vicinity of the droplet that could contain it (this short list is guaranteed to include the actual containment cell), after which the true containment cell is found from this list using ray-tracing. The overall cost (excluding the time taken to build the tree, which is a one-off cost outside the time loop) is proportional to $N_{timesteps} \log(N_{cells})$. A method of this type can be considered the default alternative approach, and is known to be applied in a range of trajectory tools. Many variants of this type of tree approach exist, and although other trees could be superior to the ADT, it is thought to be representative in performance of this family of searching methods.

To set up the comparison and reconcile the variable timestepping of the proposed method with the constant timestepping of the optimal and ADT methods, the proposed method was first computed, and the total number of timesteps and average timestep extracted from this. Both the optimal and ADT schemes were then run using the same total number of timesteps and set to use this average timestep; all methods were therefore running the same number of steps, at an equivalent level of accuracy.

The run times to calculate the example trajectory on a 160×160 unit mesh gave cost factors of 1.0, 1.93 and 6.38 for the optimal, proposed and ADT techniques, where the cost is normalised by the optimal method cost. The proposed method is therefore $3.3\times$ quicker than an alternative tree based method, but still about half the speed of the optimal method. The extra cost is mainly a result of the conditional logic needed to find the closest intersection of the velocity ray with a face of the current cell, as well as that needed to step between cells and faces through the mesh connectivity. Both the proposed and ADT methods provide a very large improvement over a naive

simple search over all cells, which is only possible to apply on tiny grids, and was not considered here.

A final test was run to determine the overhead over running BDF2 over BDF1. To do this the functional iteration loop was set to terminate at a fixed accuracy threshold ($\frac{\Delta t^{i+1}-\Delta t^i}{\Delta t^i} < 10^{-12}$), and BDF1/2 run on a mesh of 160×160. This revealed that the iterations for BDF2 required on average 7.71 cycles per timestep, whereas BDF1 used only 6.14, an overhead of 26%. However, this result is only of numerical interest; figure 6 shows that BDF2 achieves an absolute accuracy on a 160×160 mesh that is equivalent to BDF1 on a 2560×2560 mesh. This implies that for the same accuracy BDF2 will always be cheaper, as the higher mesh size cost will swamp the minor increase in iteration effort. BDF1 will also be less attractive if it is remembered that a CFD solution would also have to be run on this finer mesh, with at least a proportional cost increase.

7. Results

A more realistic case is now considered. Using an inviscid CFD tool for the underlying flow solution (note that a viscous CFD result could also have been used), figure 12(a) shows upwind droplet trajectories for an ARA wing-body at M=0.4 and an angle of attack of 3^o, for droplets of radius $10\mu m$, calculated using an inviscid 460 000 cell tetrahedral mesh (the initial condition used was that the droplet velocity matched the flow velocity in the starting cell). One particular trajectory is shown in figure 12(b), together with the intersected tetrahedra to illustrate how the timestep varies as a function of the CFD mesh cell size. The largest timestep may be many hundreds of times larger than the smallest timestep, which allows large cells to still be crossed in a single step. Despite consisting only of straight line segments the higher mesh resolution near the surface allows the trajectory to be accurate in this region, with the trajectory accuracy following that of the underlying flow solution. On standard hardware for this case BDF2 produced 812 trajectories per second.

A visual comparison revealed a difference between BDF1 and BDF2 that was barely perceptible; indeed it is worth noting that from equations (11) and (19) in the limit of low mass droplets the iterations for both schemes must recover the airflow

velocity exactly, so a significant difference would not be expected. The order of the scheme would be most important for heavy droplets, where the accuracy of the discretisation for the equation of motion assumes greater importance owing to the droplets following increasingly inertial paths.

8. Conclusions

Face intersections are often used to find streamlines, but the development here is that the method for tracking streamlines through a mesh has been combined with a method for finding the trajectory through a cell in a single implicit step. Streamlines and trajectories have been calculated using a face intersecting method on unstructured meshes and found to be of good quality. Studies show reproduction of analytical results to the expected order of accuracy across a range of mesh sizes and water catch computations for NACA 23012 are in agreement with published numerical and experimental data.

This scheme is efficient because it can traverse any cell in a single timestep, which means no time is wasted in large cells, and also finds trajectories to an accuracy commensurate with the mesh resolution in any region. Near to leading edges, where accuracy is important, small steps will be used because the cells in this region are smaller. Near the far field the larger cells will produce larger timesteps. Using an implicit formulation means that there are no restrictive stability limitations on the timestep, which is a strong advantage over any explicit method. Convex cells with an arbitrary number of faces of any shape may be used, and the searching cost within each timestep is proportional to the number of cell neighbours. This is an advantage over any method using a data tree to determine cell containment, as these typically require a search of cost $\log(N_{cells})$.

The implicit system is solved using functional iteration, which works here providing a particular factorisation is chosen. Theoretical work and experimental test have confirmed that convergence of this process depends primarily on the maximum power of Reynolds number contained in the semi-empirical relationship for the droplet drag coefficient, which should not exceed three for BDF1 or $\frac{5}{3}$ for BDF2. Fortunately all practical models for C_D use powers well below this level. In terms of cost, for the same final accuracy BDF2 is more efficient than BDF1.

References

Ascher, U., Petzhold, L., 1998. Computer methods for ordinary differential equations and diffrential-algebraic equations, 1st Edition. SIAM.

Beaugendre, H., Morency, F., Habashi, W., 2003. FENSAP-ICE's three dimensional in-flight ice accretion module: ICE3D. Journal of Aircraft 40 (2), 239–247.

Bonet, J., Peraire, J., 1991. An alternating digital tree (ADT) algorithm for 3D geomtric searching and intersection problems. International Journal for Numerical Methods in Engineering 31 (1), 1–17.

Bonfiglioli, A., Leschziner, M., 2001. A comparative study of alternative reconstruction dschemes for flux evaluation within a pressure-based fully-collocated unstructured FV scheme for incompressible flow. In: Proceedings of ECCOMAS CFD 2001. ECCOMAS.

Bourgault, Y., Habashi, W. G., Dompierre, J., Chevalier, G., 1997. A finite element Eulerian approach to the inflight icing problem. In: Proceedings of the Fifteenth International Conference on Numerical Methods in Fluid Dynamics. Springer Berlin/Heidelberg, pp. 274–279, paper 102.

Caruso, S., 1993. Lewice droplet trajectory calculations on a parallel computerAIAA paper no. AIAA-1993-0172, 31st Aerospace Sciences Meeting, Reno, NV, Jan 6-9.

Chorda, R., Blasco, J., Fueyo, N., 2002. An efficient particle-locating algorithm for application in arbitrary 2D and 3D grids. International Journal of Multiphase Flow 28, 1565–1580, doi:10.1016/S0301-9322(02)00045-9.

Darmofal, D., Haimes, R., 1996. An analysis of 3D particle path integration algorithms. Journal of Computational Physics 123, 182–195.

Gent, R., Dart, N., Cansdale, J., 2003. Aircraft icing. Philosophical Transactions: Mathematical, Physical and Engineering Sciences 358 (1776), doi: 10.1098/rsta.2000.0689.

Haegland, H., Dahlea, H., Eigestada, G., Liec, K., Aavatsmarkb, I., 2002. Improved streamlines and time-of-flight for streamline simulation on irregular grids. Advances in Water Resources 30 (4), 1027–1045, doi:10.1016/j.advwatres.2006.09.002.

Haselbacher, A., Najjar, F., Ferry, J., 2007. An efficient and robust particle-localization alogorithm for unstructured grids. Journal of Computational Physics 225, 2198–2213, doi: 10.1016/j.jcp.2007.03.018.

Kim, J., Elangovan, R., 1986. An efficient numerical computation scheme for stiff equations of droplet trajectoriesAIAA paper no. AIAA-1986-407, 24th Aerospace Sciences Meeting, Reno, NV, Jan 6-9.

Kipfer, P., Reck, F., Greiner, G., 2003. Local exact particle tracing on unstructured grids. Computer Graphics Forum 22, 133–142.

Kuang, S., Yu, A., Zou, Z., 2008. A new point-locating algorithm under three dimensional hybrid meshes. International Journal of Multiphase Flow 34, 1023–1030, doi: 10.1016/j.ijmultiphaseflow.2008.06.007.

Macpherson, G., Nordin, N., Weller, H., 2009. Particle tracking in unstructured, arbitrary polyhedral meshes for use in CFD and molecular dynamics. Communications in Numerical Methods in Engineering 25, 263–273, doi: 10.1002/cnm.1128.

Mavriplis, D., 2003. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. Tech. Rep. CR-2003-212683, NASA.

Nielson, G. M., Jung, I.-H., 1999. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domain. IEEE Transactions on Visualization and Computer Graphics 5 (4), 360–372.

Papadakis, M., Rachman, A., See-Cheuk, W., Hsiung-Wei, Y., Hung, K., Vu, G., Bidwell, C., 2007. Water droplet impingement on simulated glaze, mixed, and rime ice accretions. Tech. Rep. TM2007-213961, NASA.

Potapczuk, M. G., 1992. Lewice/e: An Euler based ice accretion code. Tech. Rep. 105389, NASA.

Putnam, A., 1961. Integratable form of droplet drag coefficient. Journal of the American Rocket Society 31, 1467–1468.

Ruff, G. A., Berkowitz, B. M., 1990. Users manual for the NASA Lewis ice accretion prediction code (LEWICE). Tech. Rep. CR-185129, NASA.

Sadarjoen, A., van Walsum, T., Hin, A., Post, F., 1994. Particle tracing algorithms for 3D curvilinear grids. Tech. Rep. Number 94-80, TU Delft.

Schafer, F., Breuer, M., 2002. Comparison of C-space and P-space particle tracing schemes on high-performance computers: accuracy and performance. International Journal for Numerical Methods in Fluids 39, 277–299, doi: 10.1002/fld.324.

Shampine, L., Gear, C., 1979. A user's view of solving stiff ordinary differential equations. SIAM Review 21 (1), 1–17.

Zhou, Q., Leschziner, M., 1999. An improved particle-locating algorithm for Eulerian-Lagrangian computations of two-phase flows in general coordinates. International Journal of Multiphase Flow 25 (5), 813–825, doi:10.1016/S0301-9322(98)00045-7.

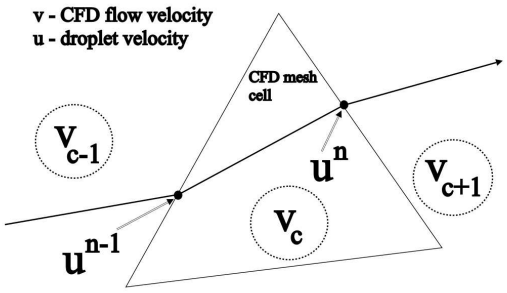
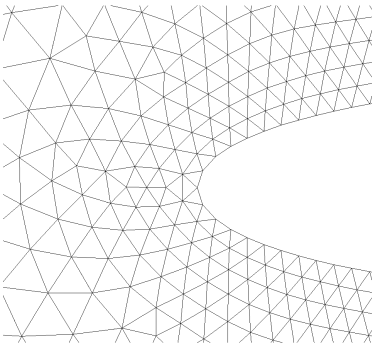
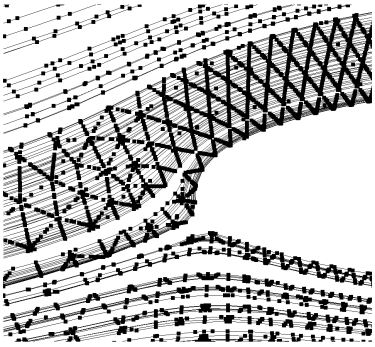


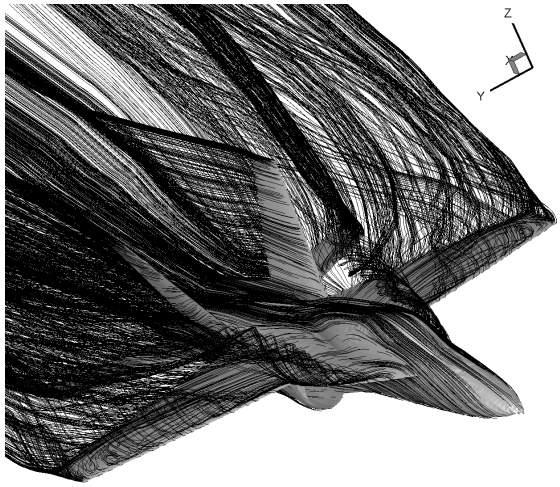
Figure 1: Flow and droplet velocities through a CFD cell



(a) NACA 0012 triangular mesh detail



(b) NACA 0012 streamlines on triangular mesh



(c) YF-17 downwind streamlines on a tetrahedral mesh

Figure 2: Finite volume streamlines

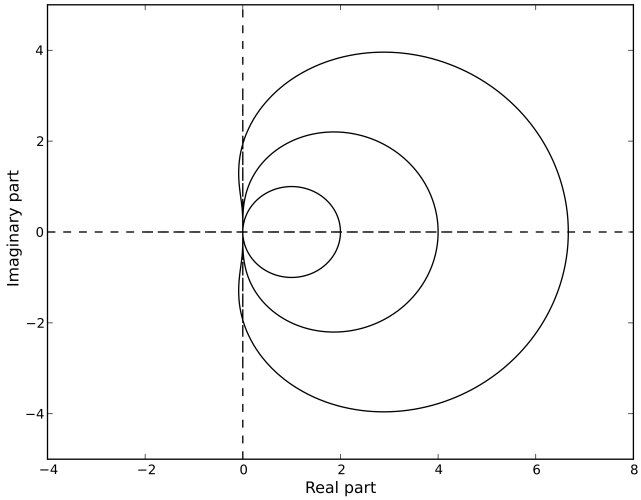


Figure 3: Stability regions for the first three backward difference formulae (stable outside contours, BDF 1→ BDF 3 moving outwards)

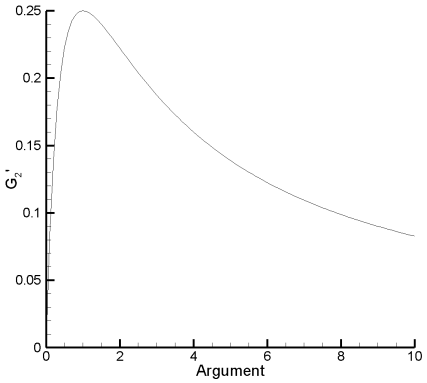
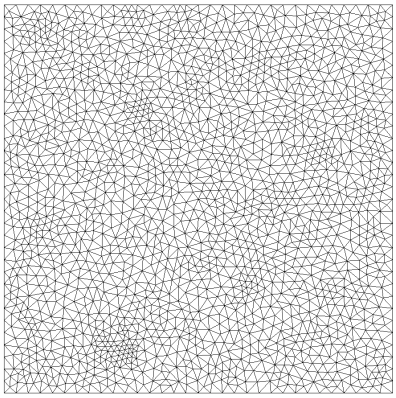
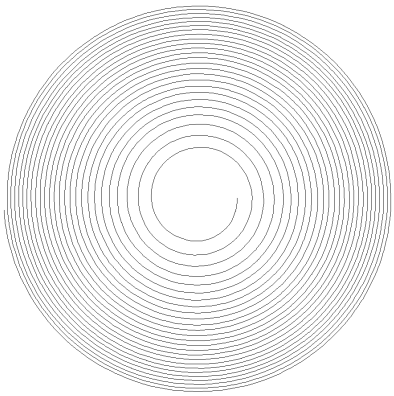


Figure 4: Plot of G_2', where 'argument' can be $|u^n - v^n|^{\alpha_p}$ or k

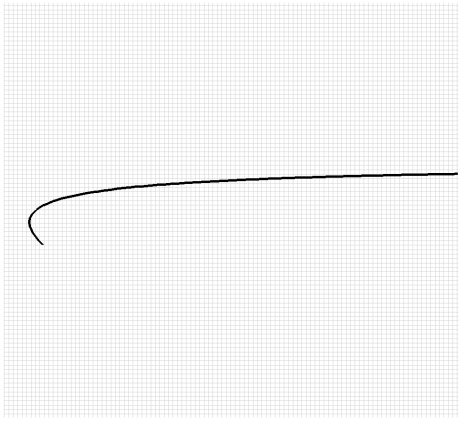


(a) Mesh (3734 triangles)

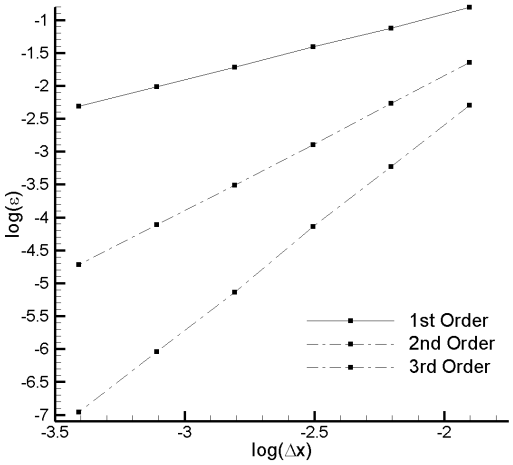


(b) Sample trajectory

Figure 5: Mesh and sample trajectory for velocity accuracy study



(a) Example trajectory



(b) Refinement study results

Figure 6: Grid refinement study for the 1^{st}, 2^{nd} and 3^{rd} order schemes

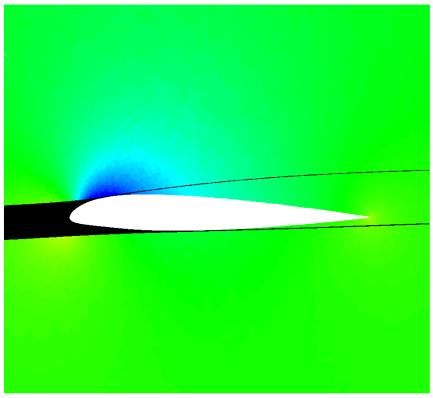
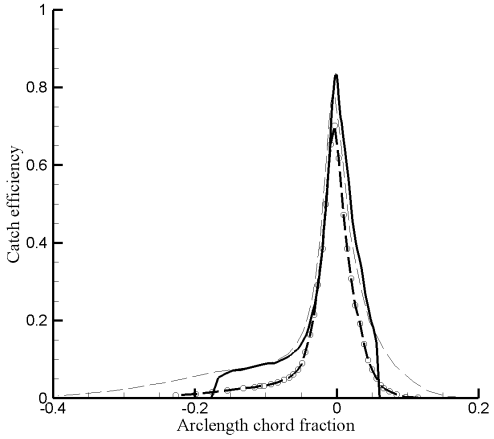
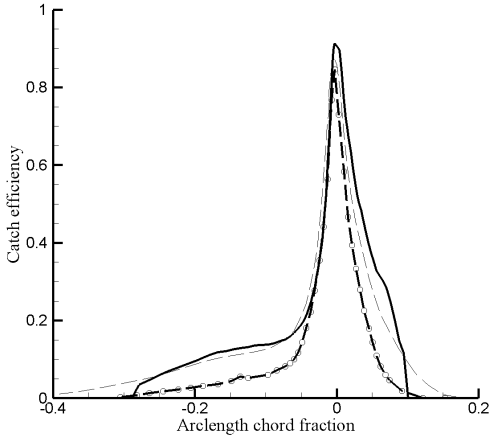


Figure 7: Impinging trajectories and C_p contours for NACA 23012 at 2.5^o, droplet diameter=154μm

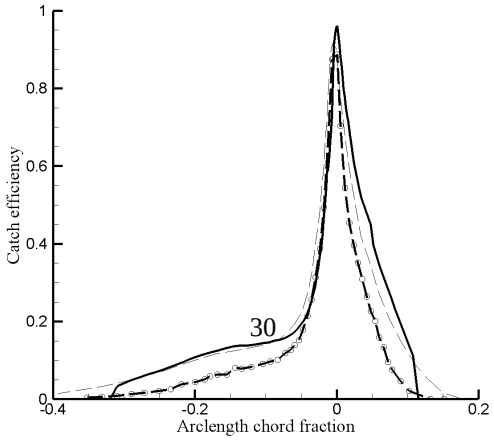


(a) Diameter=52μm



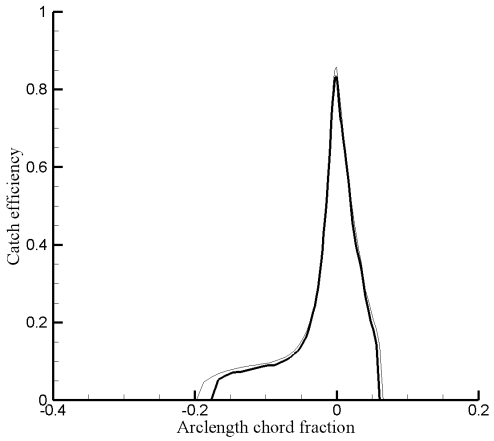
(b) Diameter=111μm

Figure 8: ARA wing-body droplet trajectories

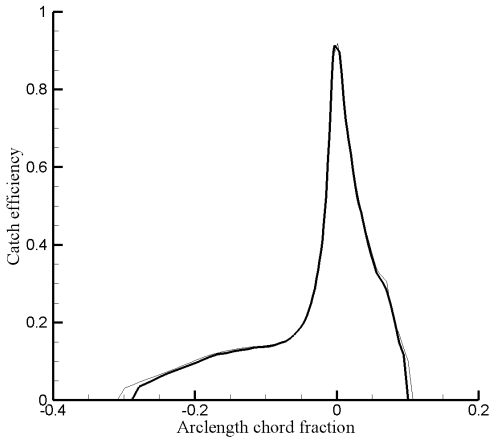


(a) Diameter=154μm

Figure 9: Catch efficiency comparisons for NACA 23012 at 2.5^o. Heavy line - current method, dashed line - Lewice, dots/heavy dashed - experiment

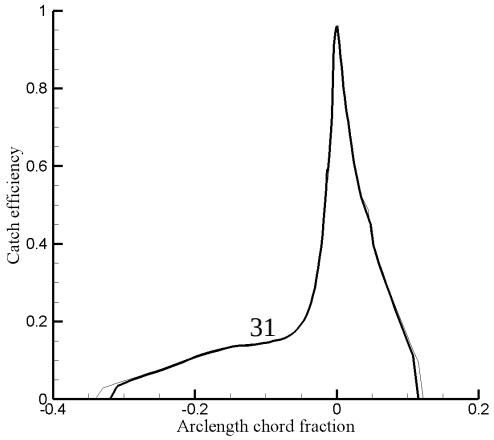


(a) Diameter=52μm



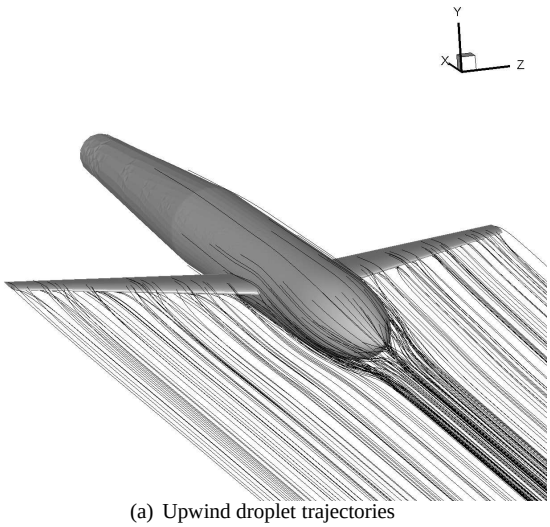
(b) Diameter=111μm

Figure 10: ARA wing-body droplet trajectories

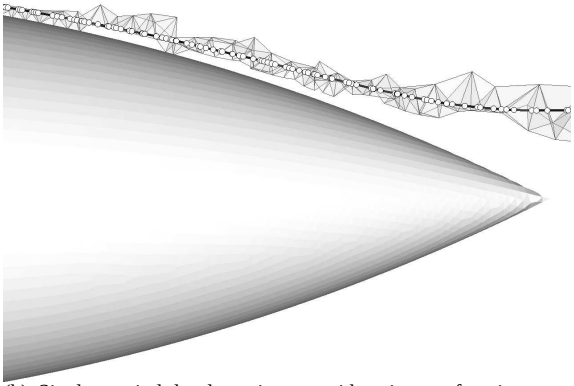


(a) Diameter=154μm

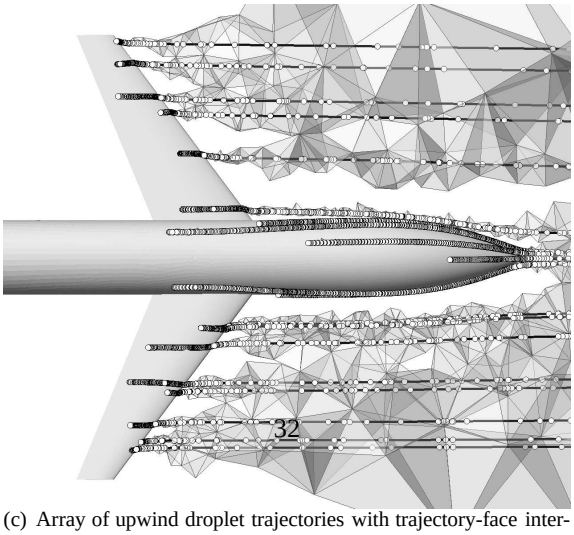
Figure 11: Catch efficiency comparisons for NACA 23012 at 2.5^o. Heavy line - constant flow velocity, light line - gradient extrapolated flow velocity



(a) Upwind droplet trajectories



(b) Single upwind droplet trajectory with trajectory-face intersections shown by dots



(c) Array of upwind droplet trajectories with trajectory-face intersections shown by dots

Figure 12: ARA wing-body droplet trajectories